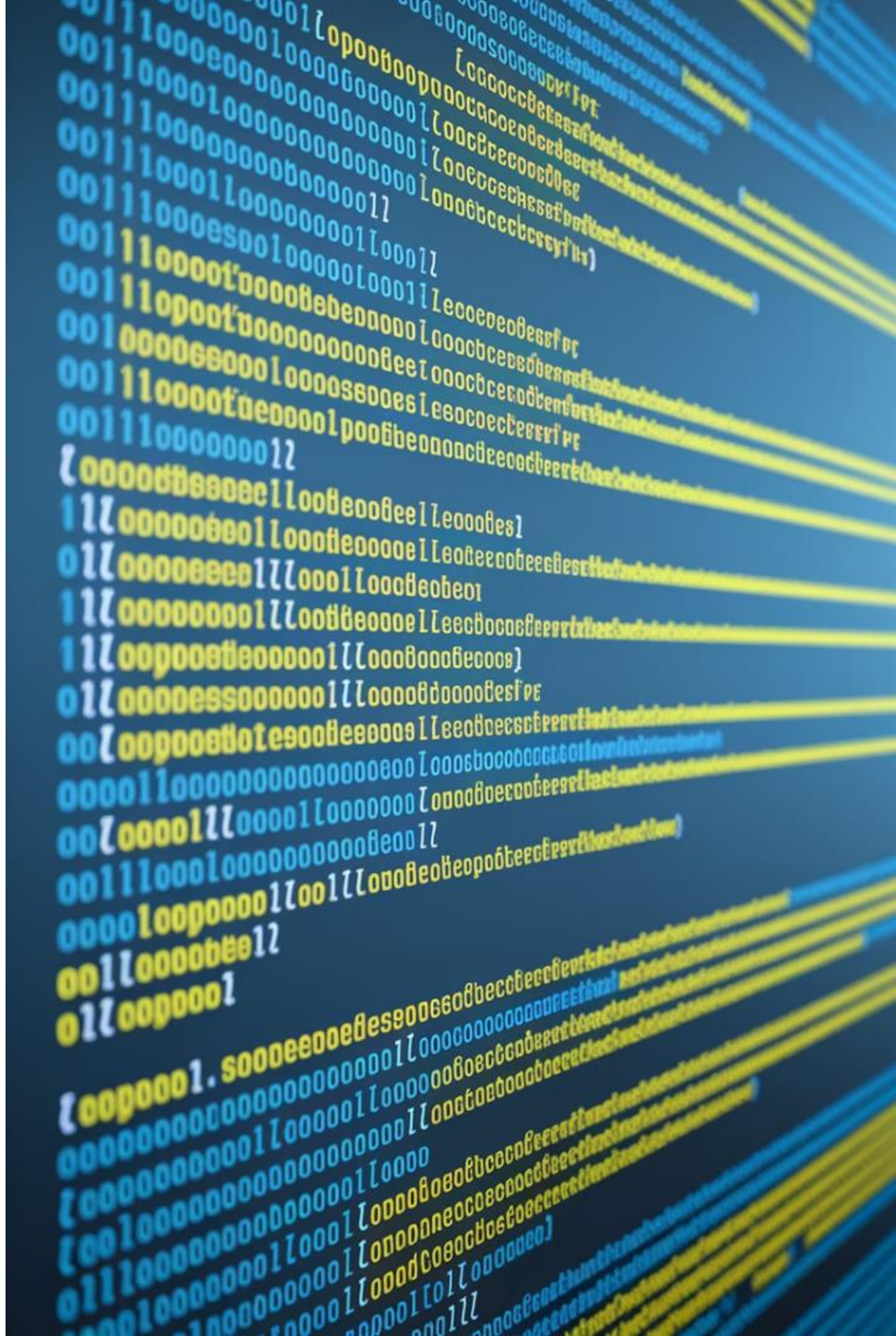
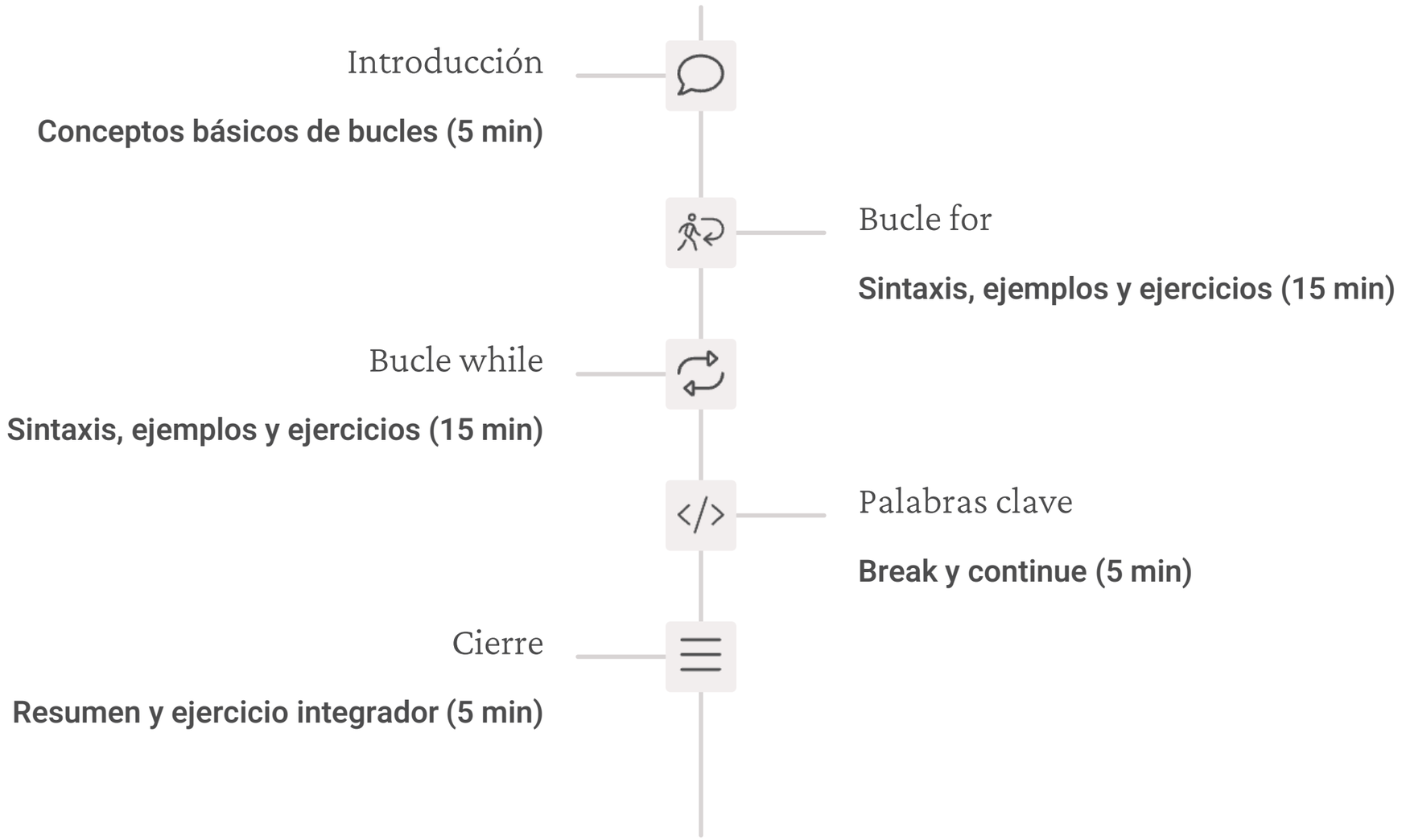
Estructuras Repetitivas en Python

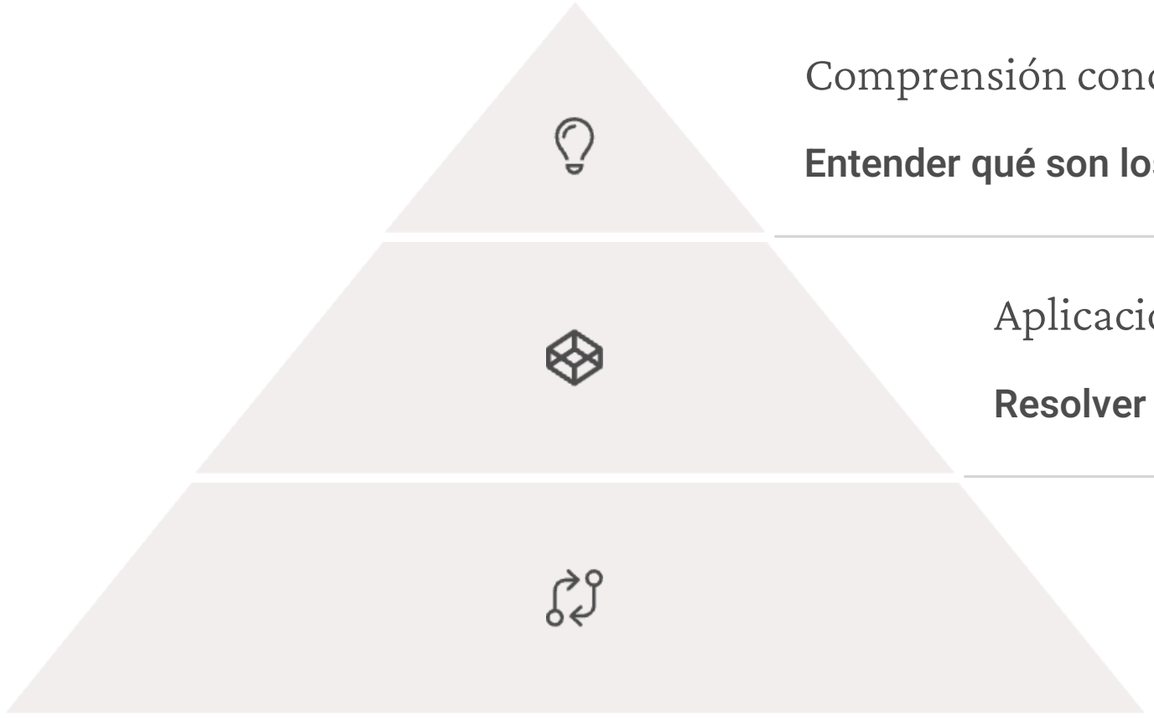
Bienvenidos a nuestra clase de microteaching sobre estructuras repetitivas en Python. Aprenderemos a automatizar tareas usando bucles for y while.



Agenda de la Clase



Objetivo General



Comprensión conceptual

Entender qué son los bucles y cómo funcionan

Aplicación práctica

Resolver problemas con ejemplos y ejercicios

Diferenciación

Identificar cuándo usar for y cuándo while

Introducción a los Bucles

¿Qué son?

Estructuras que permiten ejecutar un bloque de código múltiples veces bajo ciertas condiciones.

¿Para qué sirven?

Automatizan tareas repetitivas, evitando escribir el mismo código muchas veces.

Tipos principales

For (cantidad conocida de repeticiones) y While (condición dinámica).

El Problema

Situación

Necesitas imprimir números del 1 al 100

Solución ineficiente

Escribir 100 líneas de código

Solución eficiente

Usar bucles para automatizar la tarea

Bucle For: Definición

Estructura de control

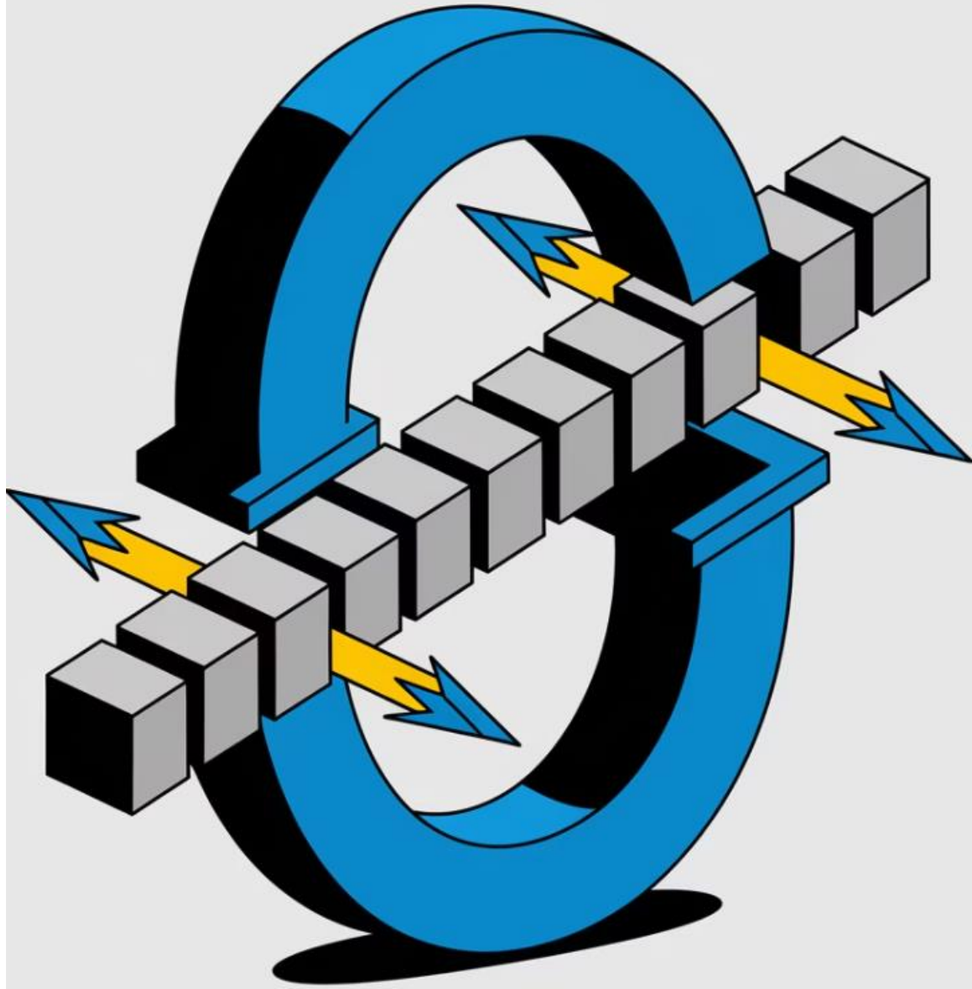
Repite un bloque de código para cada elemento en una secuencia predefinida.

Secuencias

Funciona con listas, rangos numéricos o cadenas de texto.

Uso ideal

Cuando conoces de antemano el número de iteraciones necesarias.



Sintaxis del Bucle For

Estructura básica

```
for variable in secuencia:
    # Código a repetir
```

La variable toma el valor de cada elemento en la secuencia, uno por uno.

El bloque de código indentado se ejecuta para cada elemento de la secuencia.

Función Range()

range(5)

Genera: 0, 1, 2, 3, 4

range(1, 6)

Genera: 1, 2, 3, 4, 5

range(10, 0, -2)

Genera: 10, 8, 6, 4, 2

Ejemplo de Bucle For

Código

```
# Imprimir números pares del 2 al 10
for num in range(2, 11, 2):
    print(num)
```

Salida

```
2
4
6
8
10
```

Ejercicio Rápido: For

Tarea

Escribir un for que imprima números del 5 al 1 en orden descendente.

Solución

```
for i in range(5, 0, -1):
print(i)
```

Resultado

5, 4, 3, 2, 1

Bucle While: Definición

Estructura de control

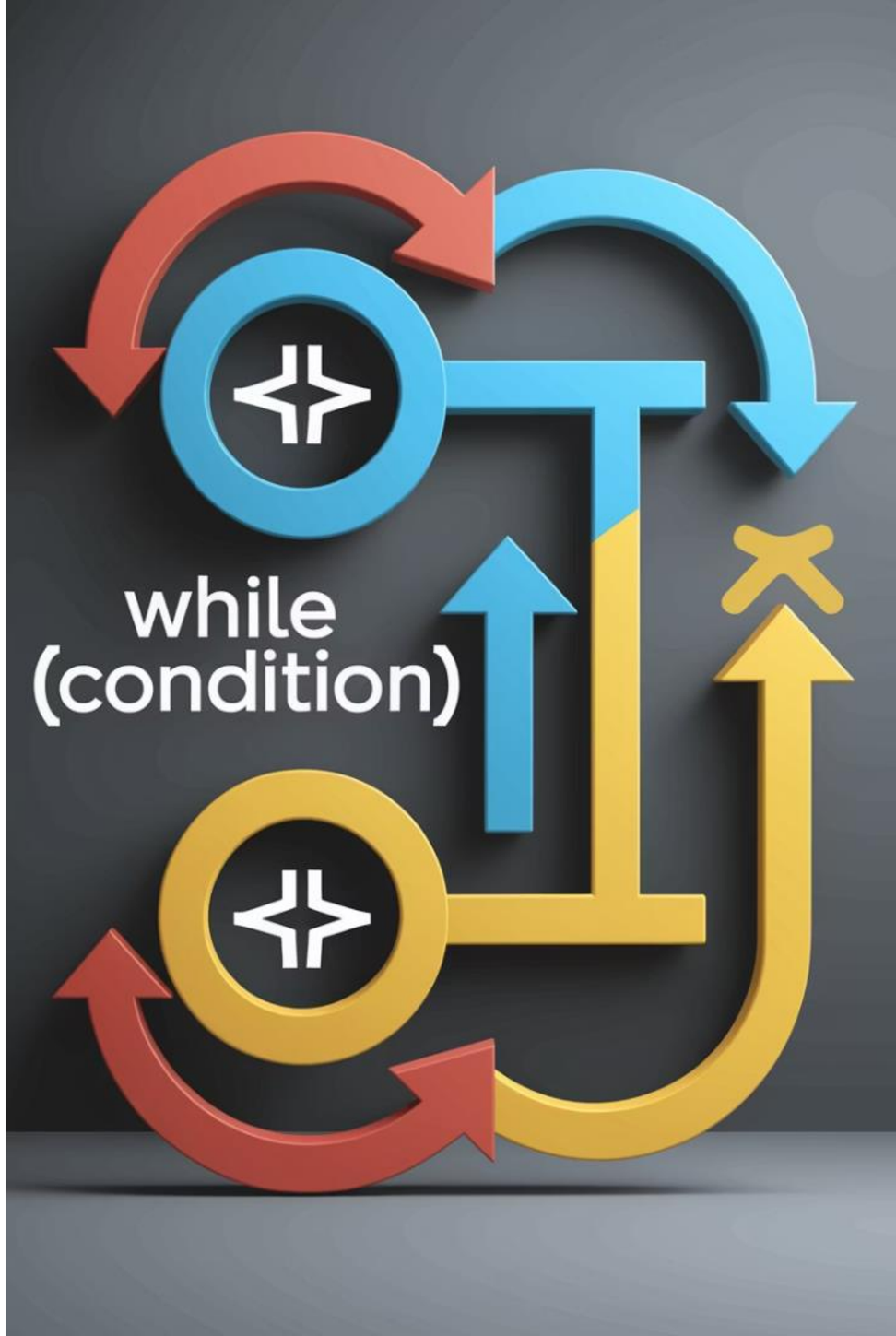
Repite un bloque de código mientras una condición lógica sea verdadera.

Uso ideal

Cuando no conoces cuántas repeticiones necesitas, pero sí la condición para continuar.

Diferencia con for

Más flexible pero requiere gestionar manualmente la condición de salida.



Sintaxis del Bucle While

Estructura básica

```
while condición:
    # Código a repetir
```

El código se ejecuta repetidamente mientras la condición sea True.

Es crucial modificar algo dentro del bucle para que la condición eventualmente sea False.

Ejemplo con Contador

Inicialización

contador = 1

Condición

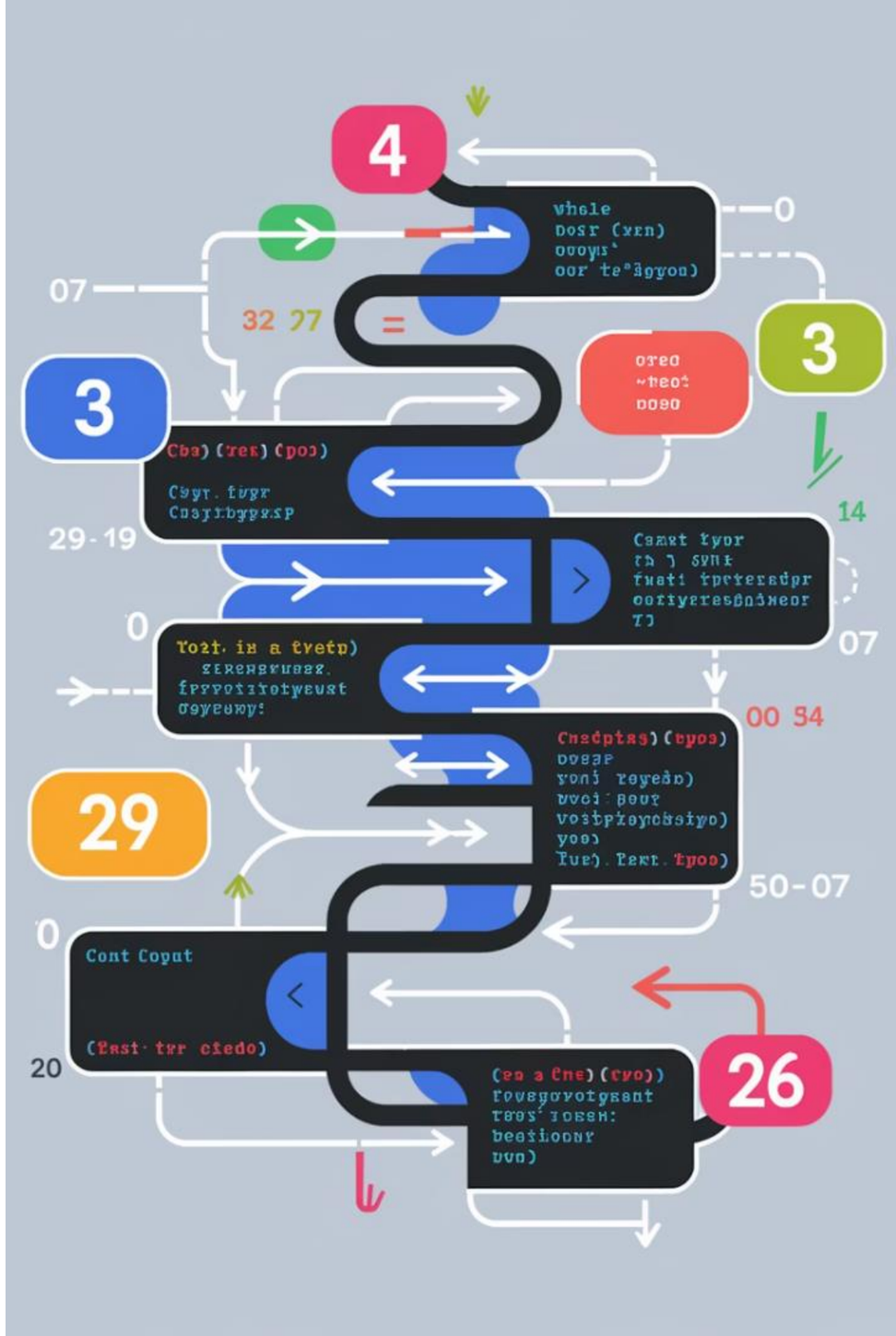
while contador <= 3:

Cuerpo del bucle

print("Vuelta:", contador)

Actualización

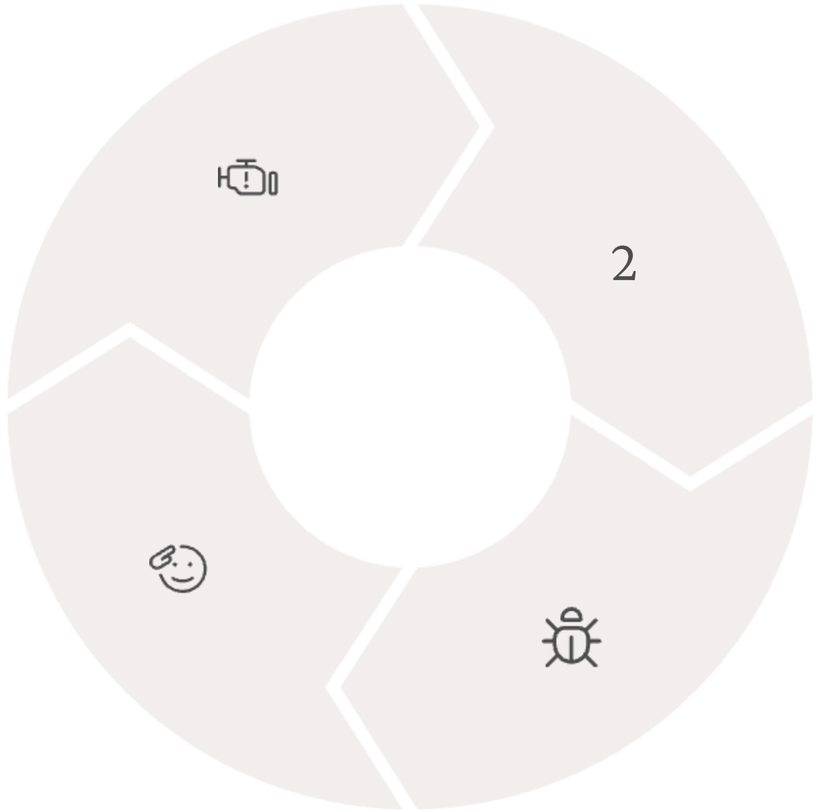
contador += 1



¡Cuidado! Bucles Infinitos

Problema

Condición siempre verdadera



Ejemplo

while True: print("¡Socorro!")

Prevención

Asegurar cambio en la condición

Consecuencia

Programa bloqueado

Ejemplo Práctico: While

Código

```
# Pedir contraseña hasta que sea correcta
clave = ""
while clave != "python123":
    clave = input("Ingresa la contraseña: ")
print("¡Acceso concedido!")
```

Explicación

El bucle continúa pidiendo la contraseña hasta que el usuario ingrese "python123".

La condición se evalúa antes de cada iteración.

Ejercicio Rápido: While

Tarea

Escribir un while que sume números hasta que el total sea mayor a 100.

2

Solución

```
total = 0
while total <= 100:
total += int(input("Ingresa un número: "))
print("Total:", total)
```

Resultado

Suma números hasta superar 100 y muestra el total.

Palabras Clave: break

Función

Detiene el bucle inmediatamente, sin importar la condición.

Útil para salir de un bucle cuando se cumple cierta condición especial.

Ejemplo

```
for i in range(1, 6):
    if i == 3:
        break
    print(i)  # Salida: 1, 2
```

Palabras Clave: continue

Función

Salta a la siguiente iteración del bucle, omitiendo el código restante.

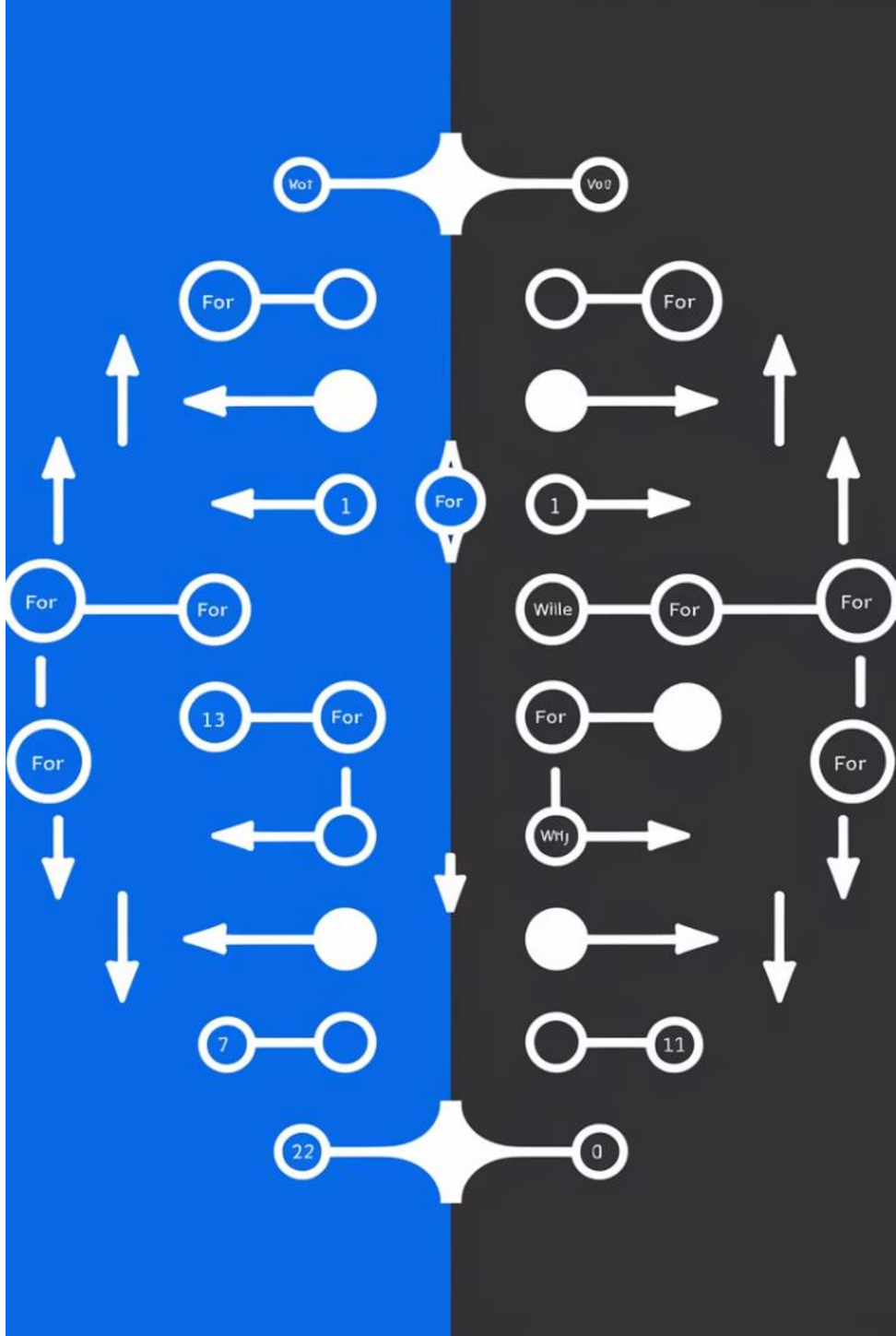
Útil para omitir ciertos valores sin detener todo el bucle.

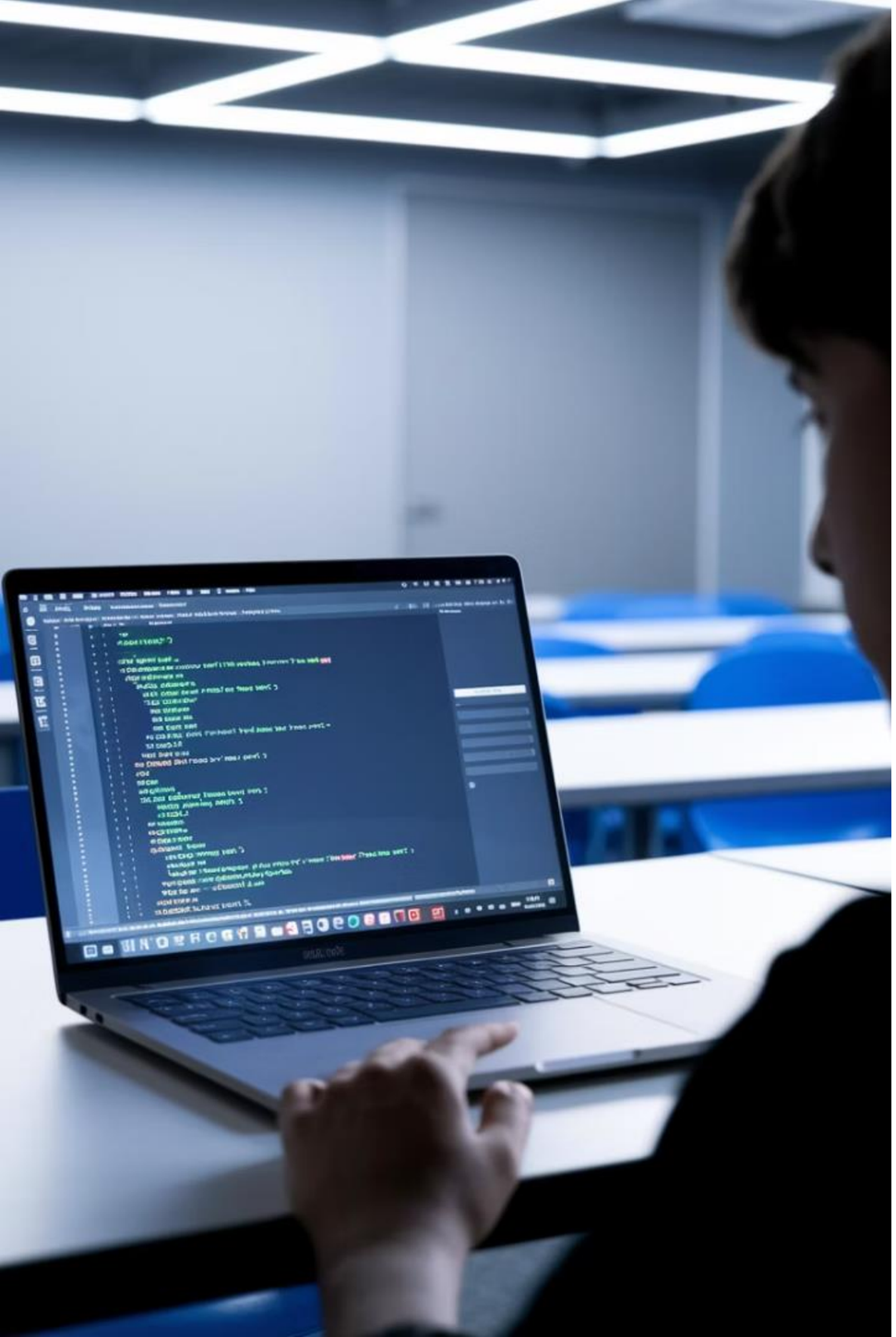
Ejemplo

```
for i in range(1, 6):
    if i == 3:
        continue
    print(i)  # Salida: 1, 2, 4, 5
```

Comparación: For vs While

Característica	For	While
Uso ideal	Número conocido de iteraciones	Condición dinámica
Control	Automático	Manual
Riesgo de bucle infinito	Bajo	Alto
Flexibilidad	Menor	Mayor





Ejercicio Integrador

Tarea

Imprimir números del 1 al 10, omitiendo el 7 y deteniendo al llegar al 9.

2

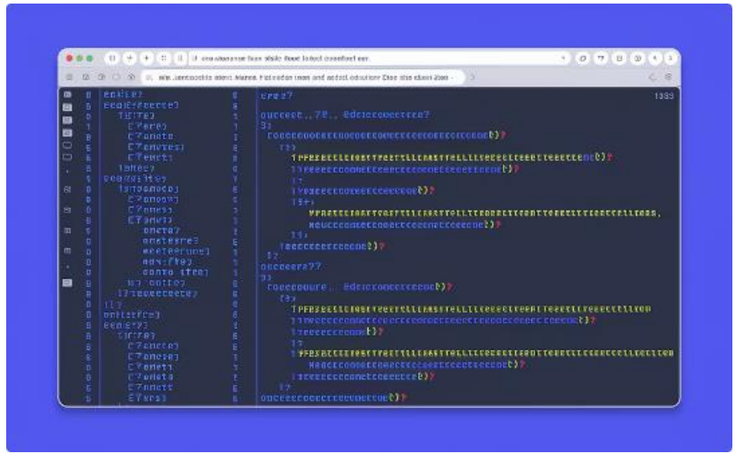
Solución

```
for i in range(1, 11):
if i == 7: continue
if i == 9: break
print(i)
```

Resultado

1, 2, 3, 4, 5, 6, 8

Herramientas para la Clase

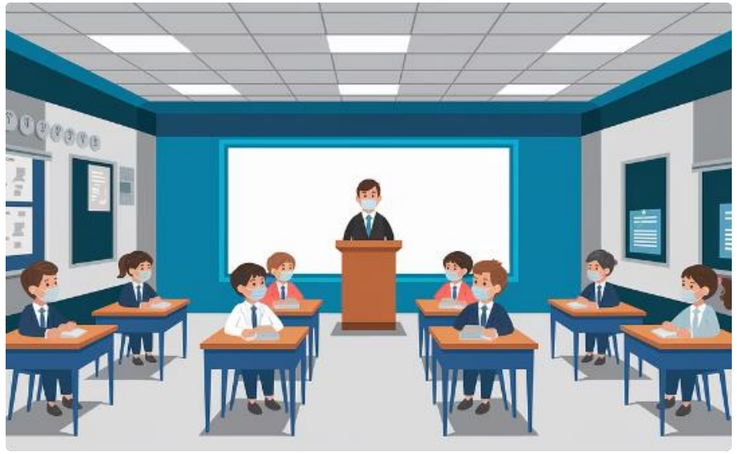


Replit

Entorno de desarrollo online para ejecutar código Python en tiempo real durante la clase.

Material de Apoyo

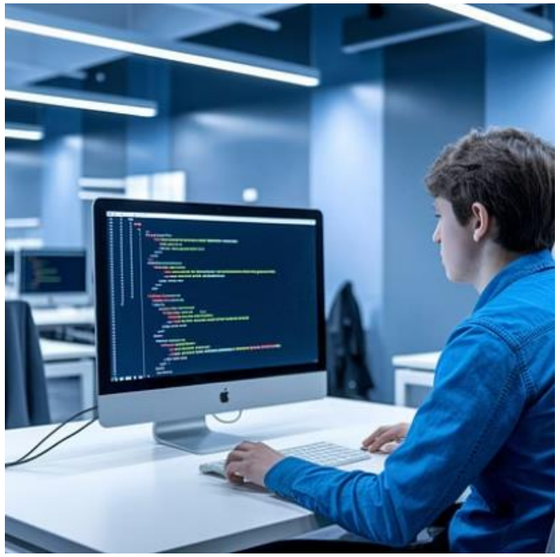
PDF con ejemplos y ejercicios adicionales para practicar después de clase.

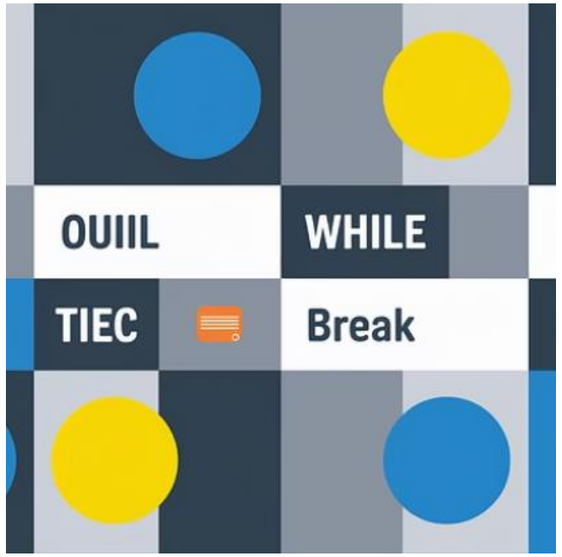


Interacción

Preguntas cada 10 minutos para verificar comprensión y resolver dudas.

Evaluación del Aprendizaje





La evaluación se realizará mediante ejercicios rápidos durante la clase y un ejercicio integrador final para verificar la comprensión de los conceptos.

Resumen: Estructuras Repetitivas

2	3	2
Tipos de bucles	Partes del bucle	Palabras clave
For y While, cada uno con propósitos específicos.	**Inicialización, condición y actualización.**	**Break y continue para control adicional.**

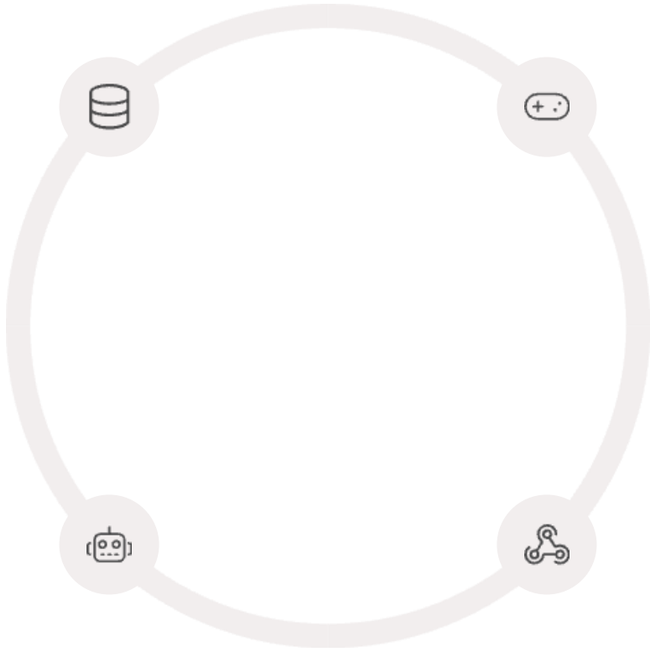
Aplicaciones Prácticas

Procesamiento de datos

Analizar listas de información automáticamente.

Automatización

Crear scripts para tareas repetitivas.



Desarrollo de juegos

Crear bucles de juego y animaciones.

Desarrollo web

Procesar formularios y datos de usuarios.

¡Gracias por tu Atención!

Recursos adicionales

Visita replit.com para practicar más ejemplos de bucles en Python.

Próxima clase

Funciones en Python: cómo crear bloques de código reutilizables.

Preguntas

¿Alguna duda sobre los bucles for o while? ¡Es el momento de preguntar!